Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Artificial Intelligence 150 (2003) 291–324

Artificial Intelligence

www.elsevier.com/locate/artint

Artificial argument assistants for defeasible argumentation

Bart Verheij

Department of Metajuridica, Universiteit Maastricht, P.O. Box 616, 6200 MD Maastricht, Netherlands

Received 7 September 2001

Abstract

The present paper discusses experimental argument assistance tools. In contrast with automated reasoning tools, the objective is not to replace reasoning, but to guide the user's production of arguments. Two systems are presented, ARGUE! and ARGUMED based on DEFLOG. The focus is on defeasible argumentation with an eye on the law. Argument assistants for defeasible argumentation naturally correspond to a view of the application of law as dialectical theory construction. The experiments provide insights into the design of argument assistants, and show the pros and cons of different ways of representing argumentative data. The development of the argumentation theories underlying the systems has culminated in the logical system DEFLOG that formalizes the interpretation of prima facie justified assumptions. DEFLOG introduces an innovative use of conditionals expressing support and attack. This allows the expression of warrants for support and attack, making it a transparent and flexible system of defeasible argumentation.

Keywords: Artificial intelligence and law; Legal reasoning; Defeasible argumentation; Argumentation software

1. Introduction

1.1. Argument assistance systems

A current goal in artificial intelligence and law is the development of experimental *argument assistance systems*. Such systems assist one or more users during a process of argumentation. A lawyer, for example, could use such a system in order to draft his

E-mail address: bart.verheij@metajur.unimaas.nl (B. Verheij).
URL: http://www.rechten.unimaas.nl/metajuridica/verheij/.

0004-3702/$ – see front matter
doi:10.1016/S0004-3702(03)00107-3

pleading in court. Such a system could be part of the lawyer's word processing package, and provide assistance, for instance, by helping the lawyer to structure his unpolished arguments, and by offering tools for analyzing the arguments. Argument assistance systems can also serve in a context of more than one user: such *argument mediation systems* can be used to keep track of diverging positions and assist in the evaluation of opinions.

More specifically, argument assistance systems are aids to drafting and generating arguments by

- administering and supervising the argument process,
- keeping track of the issues that are raised and the assumptions that are made,
- keeping track of the reasons that have been adduced for and against a conclusion,
- evaluating the justification status of the statements made, and
- checking whether the users of the system obey the pertaining rules of argument.

Marshall [26] speaks similarly of tools to support the formulation, organization and presentation of arguments.

Argument assistance systems must be distinguished from the more common automated reasoning systems. The latter automatically perform reasoning on the basis of the information in their 'knowledge base'. In this way, an automated reasoning system can do (often complex) reasoning tasks *for* the user. Argument assistance systems do not (or not primarily) reason themselves; the goal of assistance systems is not to *replace* the user's reasoning, but to *assist* the user in his reasoning process.

The different nature of argument assistance systems and automated reasoning systems has two consequences. First, argument assistance systems are more passive than automated reasoning systems. Several of their functions are implicitly available, or operate 'in the background'. For instance, the evaluation of argumentative data, such as the determination of the currently justified statements, can occur in the background, much like the automatic spelling checks of word processing systems: after each action by the user, the argument assistance system automatically updates previous evaluations.

Second, in the development of argument assistance systems, the notorious difficulties of the inherent complexities of the law (such as its open and dynamic nature) are less severe than for automated reasoning systems, since they can to a large extent be left to the user. In fact, this is a relevant incentive to develop argument assistants in the first place (cf. Leenes [20] and Section 1.2).

Other incentives for the development of argument assistance software stem from the recent research interest in dialogical theories of reasoning (see, e.g., [24]; for an insightful overview see Hage [18]), the use of computer-supported argumentation in teaching and learning (e.g., Aleven's CATO [1], related to Ashley's HYPO [2], the work of Bench-Capon et al. [5] and—not focusing on the legal domain—Suthers et al. [41] on Belvedere, [13,43]), argument analysis (Reed and Walton [35] on Araucaria, see http://www.computing.dundee.ac.uk/staff/creed/research/araucaria.html), computer-supported collaborative work focusing on argumentation (see, for instance, Shum's resource site at http://kmi.open.ac.uk/people/sbs/csca/), computer-supported and online legal mediation and dispute resolution (see, for instance, [23], and http://www.mediate.com/), knowledge management [40] and the commercial development of case management and liti-

gation support systems (see, for instance, http://www.digital-lawyer.com/digital-lawyer/resource/caseman.html).

The present paper focuses on argument assistants that have been developed with a legal context in mind, and in which the argumentation is defeasible. Defeasible argumentation is based on statements or arguments that can become defeated when they are attacked by other statements or arguments. Examples of such systems are Gordon's Pleadings Game [15], Room 5 by Loui et al. [25], Zeno by Gordon and Karacapilidis [16][1] and DiaLaw by Lodder [21]. There are many otherwise interesting and relevant systems that are not about argument assistance, not legally oriented, or not about defeasible argumentation. Examples are Nute's d-Prolog [27], NATHAN by Loui and his students (1991–1993, http://www.cs.wustl.edu/∼loui/natnathan.text), IACAS by Vreeswijk [54], Pollock's OSCAR [28,29], Tarski's World by Barwise and Etchemendy [3], and Jaspars' logic animations (http://turing.wins.uva.nl/∼jaspars/animations/).

It should be noted that the development of argument assistance systems for defeasible argumentation is still mainly in an experimental phase. A first difficulty is the lack of a canonical theory of defeasible argumentation, and more specifically of legal argumentation.[2] A second difficulty is that argument assistance systems require the design of user interfaces of a new kind. There is much to be learnt about the way arguments can be sensibly and clearly presented to the users (especially when they are defeasible), or with the way argument moves should be performed by the user. Difficulties such as these could be the cause of the striking differences between the argumentation theories and user interfaces of argument assistance systems (cf. Section 4 on related work).

Elsewhere [45,46], I have argued that even in the current experimental phase the development of argument assistance systems is relevant. I distinguished four ways in which the development of argument assistance systems is worthwhile: first, such systems can serve as *realizations* of (formal) argumentation theories, which is especially relevant because of the (well-recognized) technical difficulties of many theories; second, they are *test beds* for argumentation theories, technically, philosophically and in practice; third, argument assistance systems can be *showcases*, giving the argumentation theories more credibility; and, finally, they can be *practical aids*, with applications in, for instance, legal decision making, planning and education. Currently developed systems are already worthwhile in the first two, more theoretically oriented ways, and are starting to become so in the second two, more practically oriented ways.

In the present paper, two prototypes of argument assistants are presented, with different argumentation theories and program designs. The first is the ARGUE! system (see Section 2), the second ARGUMED based on DEFLOG (see Section 3).[3] The systems

[1] Zeno was developed in the context of a project focusing on geography, but has also been explicitly presented in the artificial intelligence and law community.

[2] For an overview of argument models in law, see [4] and the special issue of *Artificial Intelligence and Law*, Vol. 4, Nos. 3/4, 1996. For overviews of defeasible argumentation, see [33], or [8]. For an overview of nonmonotonic logics, see [12].

[3] Parts of the present paper are based on earlier publications, especially [47]. Section 1.2 is taken from [49]. The description of CUMULA (Section 2.1) is adapted from [45] and [22]. An extended version of the present paper will be published as a book [53].

can be downloaded at http://www.rechten.unimaas.nl/metajuridica/verheij/aaa/. Section 4 discusses related work. The developmental history of the systems and the main reasons for the design choices made are overviewed in Section 1.3.

Before that, we turn to the view on legal argumentation that underlies the systems' argumentation theories: legal argumentation is a kind of dialectical theory construction.

1.2. Legal argumentation as dialectical theory construction

A naive conception of the application of the law to concrete cases is that it consists in strictly following the given rules of law that match the given facts associated with a case—a conception by which a judge is turned into a *bouche de la loi* (Fig. 1).

The main problem with this view (which has become a mock image of law application that mainly serves as a starting place for discussion) is that it assumes that the rules of law and the case facts are somehow readily available. Obviously, that is not true in general. The available material is often simply not sufficiently precise and unambiguous to allow straightforward application of rules to facts. And even if the rules and facts would be given in an adequate manner, following the rules that match the case facts can be problematic. First, following the rules may not be appropriate, for instance, when a rule is not applicable because of an exception. Second it may not solve the case at all, for instance, when no relevant result follows. Third there may be several possibilities, perhaps even conflicting.

The first can occur since legal rules are generally *defeasible*. There can be exclusionary reasons or reasons against their application, for instance when applying the rule would be against its purpose.

The second is the case when there is a legal *gap*: the applicable law does not have an answer to the current case. This not only occurs on the advent of new legally relevant phenomena (such as the new legal problems as they are encountered by the rise of the Internet), but also when the law only (and often deliberately) provides a partial answer, as for instance by the use of open rule conditions, such as grievous bodily harm or fairness. An adjudicator will have to fill the gap, for instance by making new rules of classification.

The third is the case when there is a legal *ambiguity*: the applicable law provides several possible answers. This can occur by accident, for instance, when there is an unforeseen

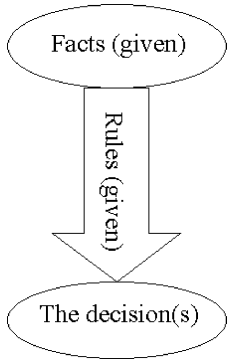

Fig. 1. A naive view of applying the law to a case.

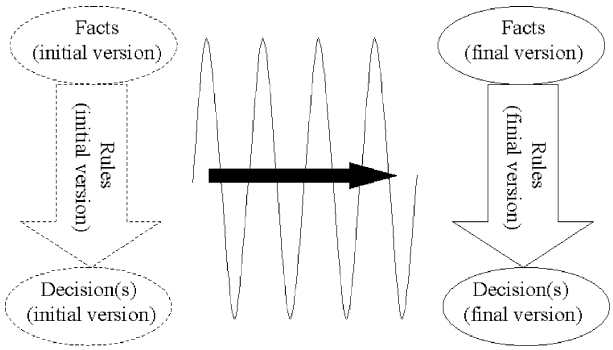

Fig. 2. Theory construction.

and unwanted conflict of rules. In a complex, man-made system such as the law, this is to be expected. Ambiguities also arise on purpose however, namely when choosing between the different possibilities is left to the discretion of the adjudicator. For instance, in the Netherlands, rules of criminal law have open rule conclusions, in the sense that they merely prescribe the maximum amount of punishment. As a result, the adjudicator can take all circumstances into account when deciding the actual amount of punishment.

Defeasibility is related to the dialectical argumentation that is so deeply entrenched in the law: every claim can at times be put to discussion. Legal gaps and ambiguities are signs of the inherent openness of the legal system. Just as defeasibility, they allow for a flexible application of the law that takes all circumstances into account, and thus can increase the system's justness.[4]

Law application can therefore best be considered as a kind of *dialectical theory construction* (Fig. 2). In such a view, applying the law to a case is a process going through a series of stages. During the process, a theory of the case, the applicable law and the consequences are progressively developed. The process starts with a preliminary theory with imperfections, such as insufficiently justified assumptions, tentative interpretations of legal sources, unduly applied rules, open issues and conflicting conclusions. During the process, the theory is gradually enhanced in order to diminish the imperfections. The process is guided by examining the preliminary theory, and by looking for reasons for and against it.

The argument assistants presented in the present paper support the dialectical theory construction needed for the application of the law to cases.

1.3. Two prototypes: ARGUE! *and* ARGUMED *based on* DEFLOG

The first argument assistant, ARGUE! (Section 2), was inspired by work on the logical system CUMULA that abstractly modeled defeasible argumentation [44]. In CUMULA, arguments (in the sense of trees of reasons and conclusions) can be defeated. The defeat

[4] Some may fear that defeasibility, gaps and ambiguities all too easily diminish legal security and equality. One asset of the legal system is that it tries to uphold legal security and equality by explicit specification, while leaving room for justness by remaining open.

of arguments results from attack by other arguments, as expressed by defeaters. A defeater indicates which set of arguments attacks which other set of arguments. CUMULA's defeaters allow the representation of several types of defeat, including defeat by parallel strengthening and by sequential weakening [44]. While building ARGUE!, it became apparent however that CUMULA (or better: the simplified version of it used for ARGUE!) was not sufficiently natural for the representation of real-life argumentation. Also the on-screen drawing of argumentative data (especially of the defeaters) seemed to be too complex for the intended users. The result was a system that was mainly interesting from a research perspective, as a realization of (and testbed for) a particular theory of defeasible argumentation. ARGUE! was first described in this way by Verheij [45].

A new approach was taken, with two starting points. First, the argumentation theory would be changed considerably. Second, the interface would become template-based. The user could perform his argumentation by filling in forms dedicated to particular argument moves.

With respect to the argumentation theory, the focus was limited to undercutting exceptions, as distinguished by Pollock [28,29]: reasons that block the connection between a reason and a conclusion. Since undercutting defeaters are of established importance for legal reasoning (see, e.g., [17,30,44], this seemed to be a natural choice. The first version of ARGUMED (ARGUMED 1.0 [46], not further discussed in the present paper) was soon replaced by the second since it had two obvious drawbacks: undercutting exceptions were not graphically represented, and it was not possible to argue about certain relevant issues, such as whether a statement was a reason or whether it was an exception. The former drawback was solved in ARGUMED 2.0 by the use of dialectical arguments, in which support by reasons and attack by undercutting exceptions were represented simultaneously. The latter led to the introduction of step and undercutter warrants. In ARGUMED 2.0, a step warrant is a kind of conditional sentence that underlies an argument step, such as 'If Peter has violated a property right, then he has committed a tort'. Undercutter warrants similarly underlie attack by an undercutting exception. An example of an undercutter warrant is the statement 'The statement that there is a ground of justification for Peter's act, is an exception to the rule that, if Peter has violated a property right, then Peter has committed a tort'. Verheij [47] gave the first presentation of ARGUMED 2.0.

ARGUMED 2.0 was evaluated by a group of ten test persons. The group was varied and consisted mostly of students and staff members of the Faculty of Law in Maastricht. They were asked to finish a test protocol containing several tasks to be performed within ARGUMED 2.0. (The test protocol is available at http://www.rechten.unimaas.nl/metajuridica/verheij/aaa/. It is however in Dutch.) The goal was to find out whether the system and its argumentation theory sufficiently spoke for themselves. For that purpose, the test protocol initially provided little information about its workings, but let the test persons find out for themselves by showing unexplained examples and by asking to reproduce argumentation samples in the system.

The test results were qualitatively evaluated. It was reassuring that some test persons almost flawlessly finished the test protocol. Most test persons indicated having enjoyed the test. The opinions about the system were reasonably positive. The opinions were more positive when the test protocol was finished more easily. The tests also showed a number of recurrent obstacles in the system and its argumentation theory. For instance, the dialectical

arguments were understood reasonably well, as long as there were no warrants involved. Not only was it hard to reproduce warrants in the system, but also their intended role in argumentation was not entirely clear to all test persons. The distinction between issues and assumptions turned out to be difficult for some test persons, especially in connection with the justification status of the statements. The template-based interface was not a complete success. For some, it was hard to relate the slots of the templates to what was happening in the argument screen. Several test persons expected that the argument screen would be mouse-sensitive, for instance, to repair small typing errors, but by trying found out that it was not.

The user evaluation of ARGUMED 2.0 inspired the design of a new user interface of the system. The result was ARGUMED based on DEFLOG (the version of ARGUMED described in this paper, see Section 3). Its user interface is based on a mouse-sensitive argument screen, in accordance with what the test persons had expected. When the user double-clicks in the argument screen, a box appears in which a statement can be typed. The right mouse button gives access to a context-sensitive menu that allows adding support for or attack against a statement. The resulting interface is very natural and easy to use (as was confirmed by another user evaluation). Apart from the better interface, the most interesting enhancement of the new version of ARGUMED is that it uses a richer and more satisfactory argumentation theory. Whereas in ARGUMED 2.0 the only kind of attack was based on undercutting exceptions, ARGUMED based on DEFLOG allows the attack of any statement. By considering the connecting arrows between statements (whether expressing support or attack) as conditional statements, warrants and undercutters found natural representations. Moreover, the new version of ARGUMED is logically more satisfactory: the evaluation of dialectical arguments corresponds exactly to the dialectical interpretations of prima facie justified assumptions in the logical system DEFLOG (see [48,52]).

The main part of this paper consists of descriptions of the systems and their argumentation theories (Sections 2 and 3). In order to illustrate the possibilities and differences, one example is used throughout the discussion of the two systems.

1.4. An example: a case of grievous bodily harm

Consider the following fictitious case of grievous bodily harm.

> There has been a pub fight, in which someone is badly hurt: according to the hospital report, the victim has several broken ribs, with complications. Someone is arrested and accused of intentionally inflicting grievous bodily harm, which is punishable with up to eight years of imprisonment, according to article 302 of the Dutch criminal code. The accused denies that he was involved in the fight. However, there are ten witnesses who claim that the accused was involved. In one precedent (referred to as precedent 1), the victim has several broken ribs, but no complications. In that precedent, the bodily harm was not considered to be grievous, and the accused was punished for intentionally inflicting ordinary bodily harm, which is punishable with up to two years of imprisonment (article 300 of the Dutch criminal code). In another precedent (referred to as precedent 2), the victim has several broken ribs with complications. In precedent 2, the accused was punished for intentionally inflicting grievous bodily harm.

The case story can give rise to interesting argumentation concerning the accused's punishability because of inflicting grievous bodily harm. In the discussion of the systems, it will be shown to what extent the relevant argumentation can be produced within each of them. In Sections 2.2 and 3.2, the example is analyzed in ARGUE! and in ARGUMED based on DEFLOG, respectively.

2. ARGUE!

2.1. Argumentation theory

The argumentation theory underlying the ARGUE! system was inspired by CUMULA [44]. CUMULA is a procedural model of argumentation with arguments and counterarguments. It is based on two main assumptions. The first assumption is that argumentation is a *process* during which arguments are constructed and counterarguments are adduced. The second assumption is that the arguments used in argumentation are *defeasible*, in the sense that whether they justify their conclusion depends on the counterarguments available at a stage of the argumentation process. If an argument no longer justifies its conclusion it is said to be defeated. The defeat of an argument is caused by a counterargument (that is itself undefeated).

For instance, if a colleague entering the room is completely soaked and tells that it is raining outside, one could conclude that it is necessary to put on a raincoat. The conclusion can be rationally justified, by giving support for it. The following argument could be given:

> A colleague entering the room is completely soaked and tells that it is raining.
> So, it is probably raining.
> So, it is necessary to put on a raincoat.

Such an argument is a reconstruction of how a conclusion can be supported.

An argument that supports its conclusion does not always justify it. For instance, if in our example it turns out that the streets are wet, but the sky is blue, the conclusion that it is necessary to put on a raincoat would no longer be justified. The argument has become *defeated*. For instance, the following argument could be given:

> The streets are wet, but the sky is blue.
> So, the shower is over.

In this case the argument that it is probably raining is defeated by the *counterargument* that the shower is over. Since the conclusion that it is probably raining is no longer justified, it can no longer support the conclusion that it is necessary to put on a raincoat.

CUMULA is a procedural model of argumentation with arguments and counterarguments. Arguments are assigned a defeat status, either undefeated or defeated. The defeat status of an argument depends on three factors:

(1) the structure of the argument;

(2) the attacks by counterarguments;
(3) the argumentation stage.

We briefly discuss each factor below. The model especially builds on the work of Pollock [28,29], Simari and Loui [39], Vreeswijk [55] and Dung [9] in philosophy and artificial intelligence, and was developed to complement work on the model of rules and reasons Reason-Based Logic (see, e.g., [17,44]).

In CUMULA, the structure of an argument (factor (1) above) is represented as in the argumentation theory of Van Eemeren and Grootendorst [10,11]. Both the subordination and the coordination of arguments are possible. It is explored how the structure of arguments can lead to their defeat. For instance, the intuitions that it is easier to defeat an argument if it contains a longer chain of defeasible steps ('sequential weakening'), and that it is harder to defeat an argument if it contains more reasons to support its conclusion ('parallel strengthening'), are investigated.

In CUMULA, which arguments are counterarguments for other arguments, that is, which arguments can *attack* other arguments (factor (2) above), is taken as the primitive notion (cf. [9]). This approach to argument defeat can be called *counterargument-triggered defeat*. Basically, an argument is defeated if it is attacked by an undefeated counterargument (cf. also [39]). This approach to argument defeat must be contrasted with *inconsistency-triggered defeat*: the primitive notion is which arguments have conflicting conclusions (as, e.g., in abstract argumentation systems [55]). In this approach to argument defeat, an argument is defeated if there is an undefeated argument with conflicting conclusion. Often the defeating argument has higher priority than the defeated argument, with respect to some priority relation on arguments.[5]

In CUMULA, so-called *defeaters* indicate which arguments are counterarguments to other arguments, that is, which arguments can defeat other arguments. In this way, CUMULA shows that the defeasibility of arguments can be fully modeled in terms of argument structure and the attack relation between arguments, independent of the underlying language. Moreover, it turns out that defeaters can be used to represent a wide range of types of defeat, as proposed in the literature, for instance, Pollock's undercutting and rebutting defeat [28]. Also some new types of defeat can be distinguished, namely defeat by sequential weakening (related to the sorites paradox; cf. [34]) and defeat by parallel strengthening (related to the accrual of reasons).

In the CUMULA model, argumentation stages (factor (3) above) represent the arguments and the counterarguments currently taken into account, and the status of these arguments, either defeated or undefeated. The model's lines of argumentation, that is, sequences of stages, give insight into the influence that the process of taking arguments into account has on the status of arguments. For instance, by means of argumentation diagrams (which give an overview of possible lines of argumentation), phenomena that are characteristic for argumentation with defeasible arguments, such as the reinstatement of arguments, are explicitly depicted. In contrast with Vreeswijk's model [55], we show how in a line

[5] I made the distinction between counterargument-triggered and inconsistency-triggered defeat in my dissertation [44]. I think that (Dung-style) counterargument-triggered defeat is philosophically the most attractive and innovative of the two approaches to argument defeat.

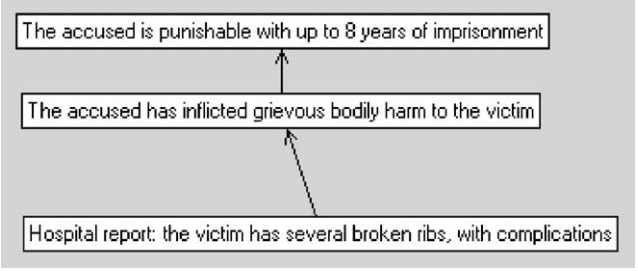

Fig. 3. A two-step argument.

of argumentation not only new conclusions are inferred ('forward argumentation', or inference), but also new reasons are adduced ('backward argumentation', or justification). In other words, CUMULA's model of the argumentation process is free, as opposed to proof-based systems (that focus on inference from a fixed set of premises) and issue-based systems (that focus on justification of a fixed set of issues): in CUMULA neither the premises nor the issues are fixed during a line of argumentation.

To summarize, CUMULA shows

(1) how the subordination and coordination of arguments is related to argument defeat;
(2) how the defeat of arguments can be described in terms of their structure, counterarguments, and the stage of the argumentation process, and independent of the logical language;
(3) how both inference and justification can be formalized in one model.

CUMULA has obvious limitations. We mention two. First, its underlying language is completely unstructured. It contains for instance no logical connectives, no quantifiers, and no modal operators. This is certainly a limitation, but one of the research objectives was to show that defeat can be fruitfully studied independently of the language. Second, the role of the rules underlying arguments is not clarified in CUMULA. This is in part due to the first limitation: the language of CUMULA does not contain a conditional or variables, by which rules would become expressible.[6]

Verheij [44] discusses the CUMULA model extensively, both informally and formally.

2.2. The grievous bodily harm example

As an illustration, it is shown how argumentation concerning the grievous bodily harm example (Section 1.4) can be represented in ARGUE!.

As a start, an argument is constructed for the conclusion that the accused is punishable with up to 8 years of imprisonment (Fig. 3). This is done by typing statements in on-screen

[6] Verheij [44] does contain a formal model in which rules play a central role, viz. Reason-Based Logic. However, the formal connection with the CUMULA model is not made. The cause of this is amongst others the very different 'flavours' of the two formalisms.

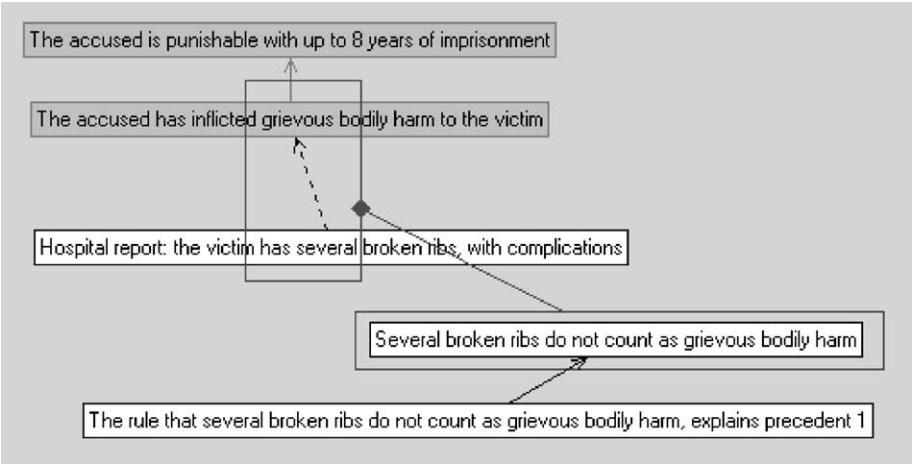

Fig. 4. Adding a defeater.

boxes and connecting the statements by drawing arrows. Here the conclusions are drawn above the reasons for them, but the user can arrange the statements at will.

In Fig. 3, all three statements are justified, as indicated by the use of white boxes. The hospital report statement is set as justified (by the user, as is indicated by a box with a different color edge), the other two are justified since there is a justifying reason for them.

Next precedent 1 is used to argue that broken ribs do not count as grievous bodily harm. The user adds the appropriate statements and draws the dedicated graphical structure that represents a defeater (Fig. 4). Here the rule that several broken ribs do not count as grievous bodily harm, which explains the precedent, is used as a counterargument against the connection between the hospital report statement and the conclusion that grievous bodily harm has been inflicted. This is an example of an undercutting defeater (cf. [28]).

The result is that the connecting arrow is no longer supporting (indicated by the dots). Therefore the conclusions that grievous bodily harm has been inflicted and that the accused is punishable are no longer justified. This is indicated by the use of gray boxes.

Finally, the accused's testimony is added as an argument attacking the conclusion that he has inflicted grievous bodily harm to the victim (Fig. 5). The result is that this conclusion is unjustified, as indicated by the crossed-out box.

For ARGUE!, the representation of the grievous bodily harm example ends here. The other relevant argumentative information cannot be represented in the right way. There are two relevant limitations of ARGUE!. First it does not allow for the representation of warrants (cf. Toulmin [42]): *that* a statement is a reason for another, cannot be the subject of further argument. Therefore the source of the punishability (the criminal code article 302) cannot be represented. Second the defeaters are not themselves statements that can be argued against. As a result, it cannot be attacked that some argument defeats another. As a result, it can for instance not be represented that the accused's testimony does not defeat the conclusion that he has inflicted grievous bodily harm to the victim, since there are ten witnesses stating that he was involved in the fight. Of course the accused's testimony can

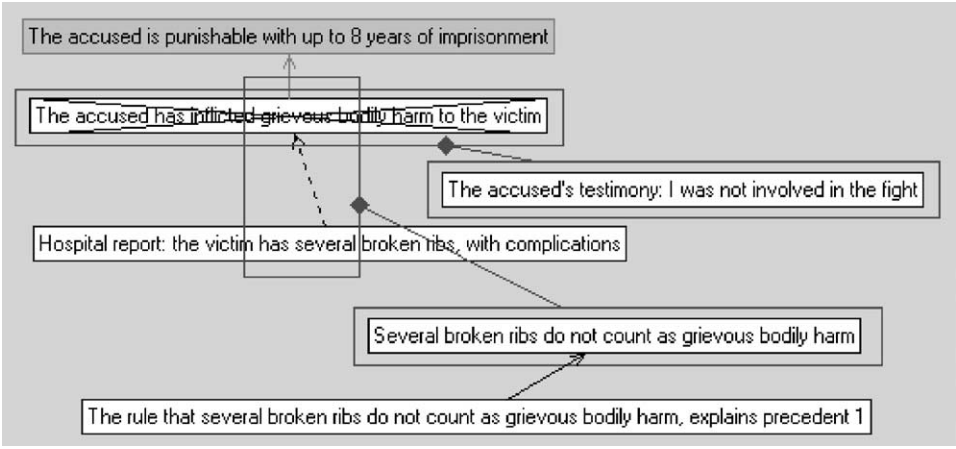

Fig. 5. A second defeater.

itself be argued against, but that would be a misrepresentation of the example: there is no reason to dispute the accused's testimony, only its defeating effect is at issue.

2.3. Program design

In the ARGUE! system, the user 'draws' his argumentation on screen. By clicking one of the buttons on the left, the user chooses the graphical mode. There are four modes. In statement mode, clicking in the drawing area shows an edit box, in which a sentence can be typed. In arrow mode, statements can be connected by arrows, indicating that a statement is a reason for another. In order to draw an arrow, the user clicks twice: first on the reason statement, second on the conclusion statement. In defeater mode, defeaters are drawn. They consist of two connected rectangles. In order to draw a defeater, the user makes two selections in the drawing area (by clicking and dragging). The first selection indicates the attacking part of the argumentative data, the second the attacked part. Only the statements and arrows that are selected are attacking or attacked, not the defeaters. In selection mode, the user can select argumentative elements in the drawing area. For instance, a statement can be moved by clicking and dragging. Statements and arrows can be deleted.

ARGUE! has a stepwise evaluation algorithm, activated by clicking the 'Evaluate (one round)' button. At each step, the current statuses of the argumentative data determine the new statuses. The basis of the evaluation is formed by the statement statuses that are set by the user. By right-clicking a statement, the user can set a statement as justified, unjustified or not evaluated.

The evaluation rules are as follows:

(1) A statement that is now set to justified or unjustified by the user, keeps its status.
(2) A statement that now has justified support, is next justified.
(3) A statement that now has no justified support and is attacked, is next unjustified.

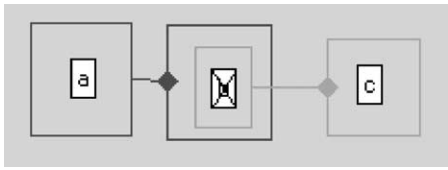

Fig. 6. An attacking statement that is attacked by another statement.

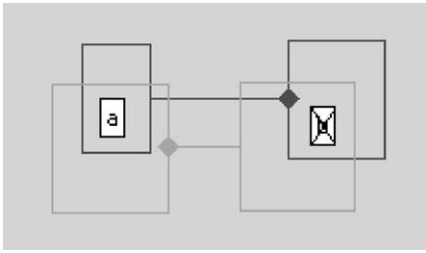

Fig. 7. Two statements attacking each other.

(4) A statement that now has no justified support and is not attacked, is next not evaluated.

A statement *has justified support* if and only if it is at the end of a supporting arrow starting at a justified statement. A statement *is attacked* if and only if it is inside the attacked rectangle of an active defeater. An arrow *is supporting* if and only if it is not inside the attacked rectangle of an active defeater. A defeater *is active* if and only if the statements in its attacking rectangle are justified and the arrows in its attacking rectangle are supporting.

The 'Jump (one round)' button activates a variant of the evaluation algorithm, in which a statement that now has no justified support and is not attacked, is next justified (instead of not evaluated). This rule has the effect that all statements are prima facie justified. The user can optionally change the selection of rules that are used when clicking either of the two buttons.

The changes of evaluation statuses are logged. It depends on the argumentative data whether new evaluations are made. Two configurations that do not lead to new evaluations (when using the 'Jump' rules) are shown in Figs. 6 and 7.

However, when in the second configuration, the statement 'a' is set to 'not evaluated', repeatedly clicking the 'Jump' button results in a loop flipping between two states: one in which both 'a' and 'b' are justified, and one in which both are unjustified. Further details are provided by Verheij [45].

3. ARGUMED based on DEFLOG

The development of ARGUE! was soon followed by a series of argument assistants with starting points that differ fundamentally from those of ARGUE!: the ARGUMED family. With respect to the program design, the starting point became that the argumentative data should be entered into the system by making argument moves instead of by drawing

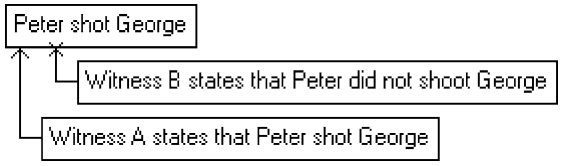

Fig. 8. Support and attack.

graphical elements. With respect to the argumentation theory, the starting point became that arguments are inherently dialectical, in the sense that support and attack go side by side and are not separated in different levels.

ARGUMED based on DEFLOG is the successor of ARGUMED 2.0 (described by Verheij [47]).[7] With respect to the program design, forms are no longer used for entering argumentative data. Instead, the screen has been made mouse-sensitive so that the user can interact directly with the argumentative data that is already shown. With respect to the argumentation theory, attack is no longer limited to undercutting exceptions, but it is possible to attack any statement. Moreover the arrows used to represent support or attack are considered as conditional statements, which allows a natural treatment of warrants and undercutters.

3.1. Argumentation theory

The argumentation theory of ARGUMED based on DEFLOG is an extension and streamlining of that of ARGUMED 2.0.

3.1.1. The structure of dialectical arguments

In ARGUMED based on DEFLOG, dialectical arguments consist of statements that can have two types of connections between them: a statement can *support* another, or a statement can *attack* another. The former is indicated by a pointed arrow between statements, the latter by an arrow ending in a cross. An example is shown in Fig. 8.

The dialectical argument consists of three elementary statements, viz. that Peter shot George, that witness A states that Peter shot George, and that witness B states that Peter did not shoot George. As is indicated, the second is a reason supporting that Peter shot George, the second a reason attacking that Peter shot George.

The expressiveness of dialectical arguments is significantly enhanced by considering the connecting arrows (of both the supporting and the attacking type) as a kind of statements, that can as such be supported and attacked. The arrow of a supporting or attacking argument step is here called the conditional underlying the step.

For instance, one could ask why A's testimony supports that Peter shot George. In Fig. 9, the statement that witness testimonies are often truthful is adduced as a reason.

The statement that witness testimonies are often truthful serves as reason why it follows from A's testimony that Peter shot George. The same statement can back the attacking argument step of B's testimony attacking that Peter shot George (Fig. 10).

[7] Verheij [46] describes ARGUMED 1.0.

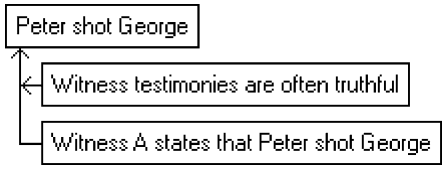

Fig. 9. Supporting that a statement is a reason for another.

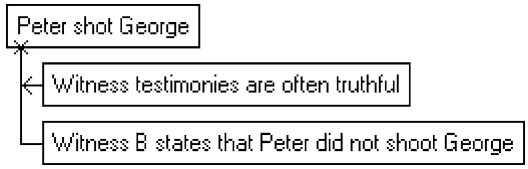

Fig. 10. Supporting that a statement is a reason against another.

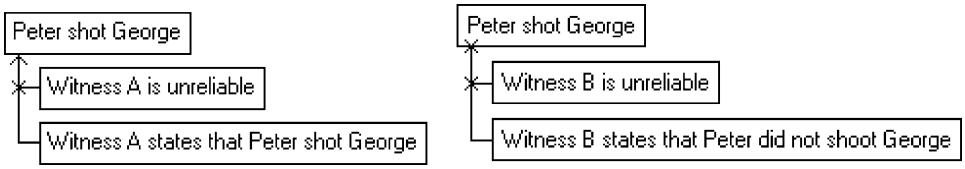

Fig. 11. Attacking that a statement is a reason.

The examples in Fig. 11 show that the connecting arrows can also be attacked.

Here the unreliability of the witnesses A and B, respectively, are adduced as reasons against the consequential effect of their testimonies.

In general, dialectical arguments are finite structures that result from a finite number of applications of three kinds of construction types:

(1) Making a statement.
(2) Supporting a previously made statement by a reason for it.
(3) Attacking a previously made statement by a reason against it.

It should be borne in mind that the types two and three consist of making two statements: one an ordinary elementary statement, viz. the reason for or against a statement, the other the special statement that the reason and the supported or attacked statement are connected, as expressed by the conditional underlying the supporting or attacking argument step.

Though dialectical arguments are here considered as the result of a finite construction, their corresponding tree structure can be virtually infinite. An example is given in Fig. 12. The dots indicate where the argument could be further extended.

The argument can be thought of as being the result of three construction steps. First the statement that Peter shot George is made, then that statement is attacked by the reason against it that Peter did not shoot George, and finally it is stated that the statement that Peter shot George is on its turn a reason against its attack. If the resulting loop is expanded as a tree (growing downward from the initial statement), the result is infinite. The relevant

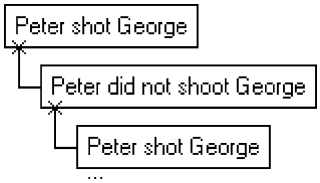

Fig. 12. An attack loop.

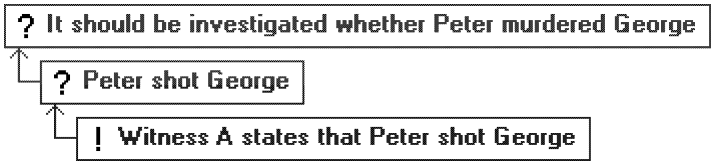

Fig. 13. An evaluated argument.

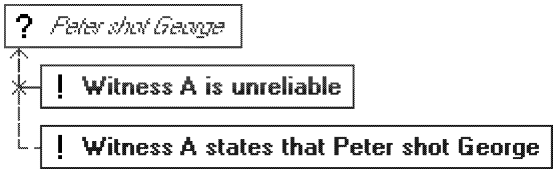

Fig. 14. An evaluated dialectical argument.

information can be finitely represented by blocking the expansion of a branch after the first recurrence of a statement, as in the figure (which was generated by the system).

3.1.2. Evaluating dialectical arguments

Dialectical arguments can be evaluated with respect to a set of prima facie justified assumptions. An example of an evaluated dialectical argument is given in Fig. 13.

Assumptions are preceded by an exclamation mark, all other statements—called issues—by a question mark. For instance, in Fig. 13, the statement that witness A states that Peter shot George is an assumption, while the other two statements shown are issues. The three shown statements are evaluated as justified, as is indicated by the dark bold font. The statement about A's testimony is justified since it is an assumption that is not attacked; the statement that Peter shot George is justified since it is supported by a justifying reason (viz. A's testimony), and similarly for the statement about the investigation. (Here and in the following the conditionals underlying argument steps are implicitly considered to be to be prima facie justified assumptions.)

The example given in Fig. 14 involves the attack of the support relation between two statements. The statements about A's testimony and unreliability are assumptions, while the statement that Peter shot George is an issue. The two assumptions are justified since they are not attacked. The statement that Peter shot George is unevaluated (as is indicated by the light italic font): it is not justified or defeated since it is an issue without justifying or defeating reason.

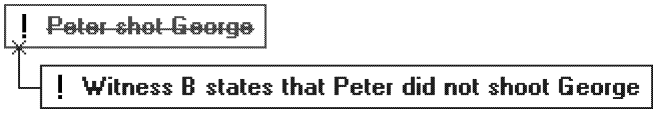

Fig. 15. A defeated assumption.

An example of a dialectical argument in which a statement is defeated is as in Fig. 15. Here the statement that Peter shot George is an assumption. Just like all assumptions, it is prima facie justified. However in the argument shown it is actually defeated (as is indicated by the bold struck-through font) since it is attacked by the reason against it that witness B states that Peter did not shoot George.

The evaluation of dialectical arguments with respect to a set of prima facie justified assumptions is naturally constrained as follows:

(1) A statement is *justified* if and only if
   (a) it is an assumption, against which there is no defeating reason, or
   (b) it is an issue, for which there is a justifying reason.
   A statement is *defeated* if and only if there is a defeating reason against it.
(2) A reason is *justifying* if and only if the reason and the conditional underlying the corresponding supporting argument step are justified.
(3) A reason is *defeating* if and only if the reason and the conditional underlying the corresponding attacking argument step are justified.

It is a fundamental complication of dialectical argumentation that a dialectical argument can have any number of evaluations with respect to a set of prima facie justified assumptions: there can be no evaluation, or one, or several.

Assuming as we do that statements cannot be both justified and defeated, the argument whether Peter shot George shown in Fig. 8 has no evaluation with respect to the testimonies by A and B as assumptions. That the argument has no evaluation is seen as follows. Since both assumptions are not attacked they must be justified in every evaluation. But then A's testimony would require that it is justified that Peter shot George, while at the same time B's testimony would require that it is defeated that Peter shot George. This is impossible.

An example of a dialectical argument with two evaluations is the looping argument discussed in Fig. 16. The argument has two prima facie justified assumptions, viz. that Peter shot George and that Peter did not shoot George. The assumptions attack each other. In one evaluation, it is justified that Peter shot George, thus making it defeated that Peter did not shoot George, while in the other evaluation it is the other way around.

Note that the existence of the two evaluations is possible because the loop of attacks consists of an even number of statements. An odd length loop of attacks can cause that there is no evaluation. Two examples are shown in Fig. 17. In the example on the left, there are three assumptions. The first is that A says that he is lying. The second (represented by the supporting arrow) is that A's saying that he is lying supports that he is lying. The third (representing by the attacking arrow) is that when A is lying A's saying that he is lying

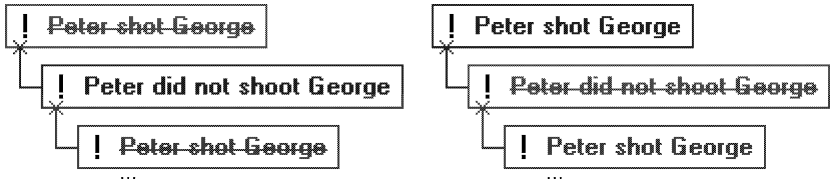

Fig. 16. An example with two evaluations.

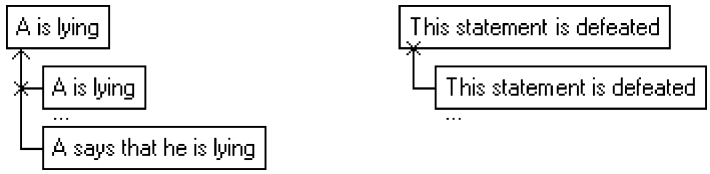

Fig. 17. Two examples in which there is no evaluation.

provides no support for A's lying. By reasoning that is well known from all variants of the liar's paradox it follows that there is no evaluation.[8] The example on the right with a self-attacking assumption is similar.[9]

3.1.3. DEFLOG*: on the logical interpretation of prima facie justified assumptions*

The ideas on dialectical argumentation discussed above can be made formally precise in terms of the logical system DEFLOG [48,52].

The dialectical interpretation of theories. DEFLOG's starting point is a simple logical language with two connectives $\times$ and $\rightsquigarrow$. The first is a unary connective that is used to express the defeat of a statement, the latter is a binary connective that is used to express that one statement supports another. When φ and ψ are sentences, then $\times\varphi$ (φ's so-called *dialectical negation*) expresses that the statement φ is defeated, and $(\varphi \rightsquigarrow \psi)$ that the statement φ supports the statement ψ. Attack, denoted as $\rtimes$, is defined in terms of these two connectives: $\varphi \rtimes \psi$ is defined as $\varphi \rightsquigarrow \times\psi$, and expresses that the statement φ attacks the statement ψ, or equivalently that φ supports the defeat of ψ. When p, q, r and s are elementary sentences, then $p \rightsquigarrow (q \rightsquigarrow r)$, $p \rightsquigarrow \times(q \rightsquigarrow \times r)$ and $(p \rightsquigarrow q) \rightsquigarrow (p \rightsquigarrow \times(r \rightsquigarrow s))$ are some examples of sentences. (For convenience, outer brackets are omitted.)

[8] Assume that there is an evaluation. When the statement that A is lying were justified in the evaluation, it would have to be justified by A's saying that he is lying. However, that is impossible since the statement that A is lying then attacks the supporting connection. The statement that A is lying cannot be defeated either since it is not attacked. But when the statement that A is lying is neither justified nor defeated in the evaluation, A's saying that he is lying justifies that A is lying, contradicting that it is not justified that A is lying. By reductio ad absurdum it follows that there is no evaluation.

[9] Note that for DEFLOG the statement 'This statement is defeated' is taken as an elementary statement, just like 'John is a thief' or 'p'. DEFLOG's language does not include a demonstrative 'this' nor does it contain a predicate 'is defeated'.

The central definition of DEFLOG is its notion of the *dialectical interpretation* of a theory. Formally, DEFLOG's dialectical interpretations of theories are a variant of Reiter's extensions of default theories [36], Gelfond and Lifschitz's stable models of logic programming [14], Dung's stable extensions of argumentation frameworks [9], and Bondarenko et al.'s stable extensions of assumption-based frameworks [7].[10]

A theory is any set of sentences. A theory represents a set of prima facie justified assumptions. When a theory is dialectically interpreted, all sentences in the theory are evaluated, either as justified or as defeated. (This is in contrast with the interpretation of theories in standard logic, where all sentences in an interpreted theory are assigned the same positive value, namely true, for instance, by giving a model of the theory.)

An assignment of the values justified or defeated to the sentences in a theory gives rise to a dialectical interpretation of the theory, when two conditions are fulfilled. First, the justified part of the theory must be conflict-free. Second, the justified part of the theory must attack all sentences in the defeated part. Formally the definitions are as follows.

(i) Let T be a set of sentences and φ a sentence. Then T *supports* φ when φ is in T or follows from T by the repeated application of $\rightsquigarrow$-Modus ponens (from $\varphi \rightsquigarrow \psi$ and φ, conclude ψ). T *attacks* φ when T supports $\times\varphi$.

(ii) Let T be a set of sentences. Then T is *conflict-free* when there is no sentence φ that is both supported and attacked by T.

(iii) Let Δ be a set of sentences, and let J and D be subsets of Δ that have no elements in common and that have Δ as their union. Then (J, D) *dialectically interprets* the theory Δ when J is conflict-free and attacks all sentences in D. The sentences in J are the *justified statements* of the theory Δ, the sentences in D the *defeated statements*.

(iv) Let Δ be a set of sentences and let (J, D) dialectically interpret the theory Δ. Then $(\mathrm{Supp}(J), \mathrm{Att}(J))$ is a *dialectical interpretation* or *extension* of the theory Δ. Here $\mathrm{Supp}(J)$ denotes the set of sentences supported by J, and $\mathrm{Att}(J)$ the set of sentences attacked by J. The sentences in $\mathrm{Supp}(J)$ are the *justified statements* of the dialectical interpretation, the sentences in $\mathrm{Att}(J)$ the *defeated statements*.

Note that when (J, D) dialectically interprets Δ and $(\mathrm{Supp}(J), \mathrm{Att}(J))$ is the corresponding dialectical interpretation, J is equal to $\mathrm{Supp}(J) \cap \Delta$, and D to $\mathrm{Att}(J) \cap \Delta$. It is convenient to say that a dialectical interpretation $(\mathrm{Supp}(J), \mathrm{Att}(J))$ of a theory Δ *is specified by* J.

The examples discussed in Sections 3.1.1 and 3.1.2 can be used to illustrate these definitions. Let the sentence s express Peter's shooting of George, a A's testimony, b

[10] See [48,52] for a discussion of relations between the formalisms mentioned. To guide intuition, the following may be useful. An attack (A, B) (as in [9]) would in DEFLOG be expressed by a sentence $\mathrm{A} \rightsquigarrow \times\mathrm{B}$. A default $p : q/r$ (as in [36]) would in DEFLOG be translated to two conditionals, viz. $p \rightsquigarrow r$ and $\neg q \rightsquigarrow \times(p \rightsquigarrow r)$. The second says that the former is defeated in case of $\neg q$. This corresponds to the intuition underlying the default that r follows from p as long as q can consistently be assumed. (Note however that the properties of ordinary negation $\neg$ are not part of DEFLOG.) A rule in logic programming $p \rightsquigarrow q, {\sim}r$ where $\sim$ is negation as failure, corresponds in DEFLOG to two conditionals, viz. $q \rightsquigarrow p$ and $r \rightsquigarrow \times(q \rightsquigarrow p)$. The second says that $q \rightsquigarrow p$ is defeated in case of r. This corresponds to the intuition underlying the program rule that p follows when q is proven, while r is not.

B's testimony, t the truthfulness of testimonies, u A's unreliability, and i the obligation to investigate. Then the example shown in Fig. 13 corresponds to the three-sentence theory $\{a, a \rightsquigarrow s, s \rightsquigarrow i\}$. The arrows in the figure correspond to the two conditional sentences. The theory has a unique extension in which the three assumptions in the theory are justified. In the extension, two other statements are justified, viz. s and i. The example in Fig. 15 corresponds to the theory $\{b, b \rightsquigarrow \times s, s\}$. The arrow ending in a cross in the figure corresponds to the sentence $b \rightsquigarrow \times s$. The theory is not conflict-free, but has a unique extension in which b and $b \rightsquigarrow \times s$ are justified, while s is defeated. In the extension, there is one other interpreted statement, viz. $\times s$, which is justified. The example of Fig. 9 corresponds to the theory $\{a, t, t \rightsquigarrow (a \rightsquigarrow s)\}$. In its unique extension, all statements of the theory are justified, and in addition $a \rightsquigarrow s$ and s. The example of Fig. 14 corresponds to the theory $\{a, u, u \rightsquigarrow \times(a \rightsquigarrow s)\}$. In its unique extension, $a \rightsquigarrow s$ is defeated and s is not interpreted (i.e., neither justified nor defeated). Note that the theory $\{a, u, u \rightsquigarrow \times(a \rightsquigarrow s), a \rightsquigarrow s\}$ has the same unique extension, but is not conflict-free.

DEFLOG's logical language only uses two connectives, viz. $\rightsquigarrow$ and $\times$. Notwithstanding its simple structure, many central notions of dialectical argumentation can be analyzed in terms of it. For instance, it is possible to define an inconclusive conditional (a conditional for which the consequent does not always follow when its antecedent holds) in terms of DEFLOG's defeasible conditional (that is defeasible in the same way as any other statement). Other examples of DEFLOG's expressiveness are Toulmin's warrants and backings [42] and Pollock's undercutting and rebutting defeaters [28]. Verheij [48] discusses how to express these notions.

Theories without and with several extensions. The examples of theories discussed above all had a unique extension. Several were examples of the following general property: a conflict-free theory always has a unique extension, namely the extension specified by the theory itself. The simplest theory that is not conflict-free with a unique extension is $\{p, \times p\}$. In its extension, p is defeated and $\times p$ justified. Other important examples of theories that are not conflict-free, but do have a unique extension are $\{p, q, q \rightsquigarrow \times p\}$ and $\{p, q, r, q \rightsquigarrow \times p, r \rightsquigarrow \times q\}$. In the former theory, the statement that p is attacked by the statement that q. In its unique extension, q and $\times p$ are justified and p is defeated. In the latter theory, a superset of the former, in addition to q's attack of p, r attacks q. In its unique extension, p, $\times q$ and r are justified, and q is defeated. The theories together provide an example of *reinstatement*: a statement is first defeated, since it is attacked by a counterargument, but becomes justified by the addition of a counterattack, that is, an attack against the counterargument. Here p is reinstated: it is first successfully attacked by q, but the attack is then countered by r attacking q.

There are however also theories with no or with several extensions:

(i) The three theories $\{p, p \rightsquigarrow \times p\}$, $\{p, p \rightsquigarrow q, \times q\}$ and $\{p_i \mid i \text{ is a natural number}\} \cup \{p_j \rightsquigarrow \times p_i \mid i \text{ and } j \text{ are natural numbers, such that } i < j\}$ lack extensions. For the latter theory, this can be seen as follows. Assume that there is an extension E in which for some natural number n p_n is justified. Then all p_m with $m > n$ must be defeated in E, for if such a p_m were justified, p_n could not be justified. But that is impossible, for the defeat of a p_m with $m > n$ can only be the result of an attack by a justified

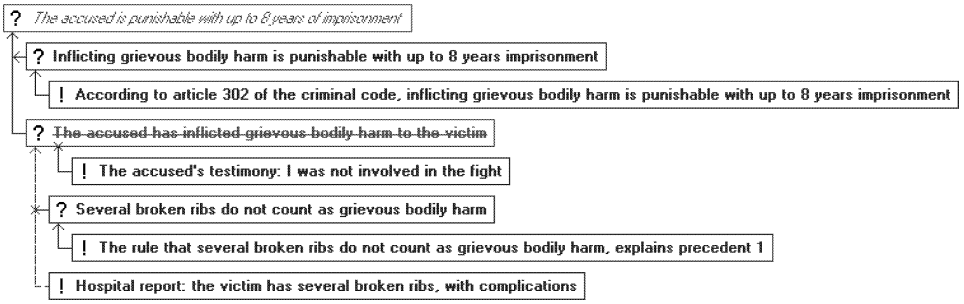

Fig. 18. A defeated reason.

$p_{m'}$ with $m' > m$. As a result, no p_i can be justified in E. But then all p_i must be defeated in E, which is impossible since the defeat of a p_i can only be the result of an attack by a justified p_j with $j > i$. (Note that any *finite* subset of the latter theory has an extension, while the whole theory does not. This shows a 'non-compactness' property[11] of extensions.)

(ii) The three theories $\{p, q, p \rightsquigarrow \times q, q \rightsquigarrow \times p\}$, $\{p_i, p_{i+1} \rightsquigarrow \times p_i \mid i \text{ is a natural number}\}$ and $\{\times^i p \mid i \text{ is a natural number}\}$ have two extensions. Here $\times^i p$ denotes, for any natural number i, the sentence composed of a length i sequence of the connective $\times$, followed by the constant p. (Note that each finite subset of the latter theory has a unique extension, showing another non-compactness property.)

3.2. The grievous bodily harm example

ARGUE! could not represent all argumentation concerning the grievous bodily harm example of Section 1.4. ARGUE! allowed the attack of statements, but could not deal with the warrants underlying argument steps. In ARGUMED based on DEFLOG, it is possible to argue about step warrants. For instance, returning to the argumentation of Fig. 5, it can be asked why it is the case that the statement that the accused has inflicted grievous bodily harm to the victim, is a reason for the conclusion that the accused is punishable with up to 8 years of imprisonment? Fig. 18 shows the argument why: in general, inflicting grievous bodily harm is punishable with up to 8 years imprisonment, and this is the case because of article 302 of the criminal code.

Note the fundamentally different ways in which attack is represented in Figs. 5 and 18: in the former representation, attack is a relation between argument structures, whereas in the latter representation, attack is a relation between statements. In Fig. 18, the conclusion that the accused is punishable is not justified since the only reason for it (the inflicting of grievous bodily harm) is not justified, even defeated by the accused's testimony.

In the case story, there is further information that makes the accused's testimony non-defeating: the testimonies of 10 pub visitors that the accused was involved in the fight. Fig. 19 shows how the argument is extended to incorporate this information. Still there is

[11] A property P of sets is called compact if a set S has property P whenever all its finite subsets have the property. Cf. the compactness of satisfiability in first-order predicate logic.

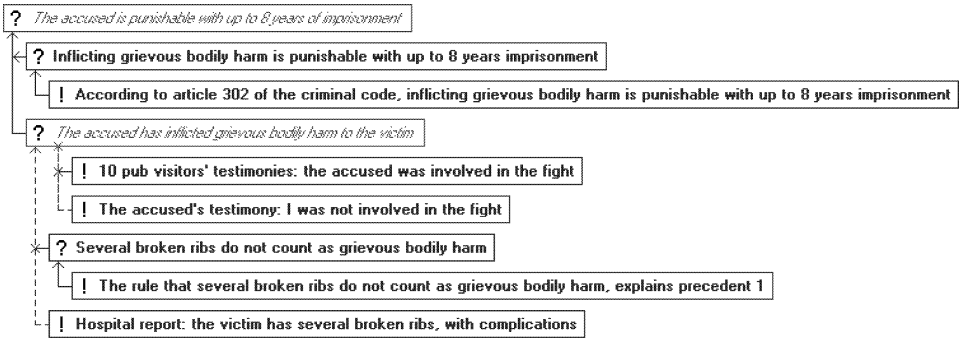

Fig. 19. A reason that is neither justified nor defeated.

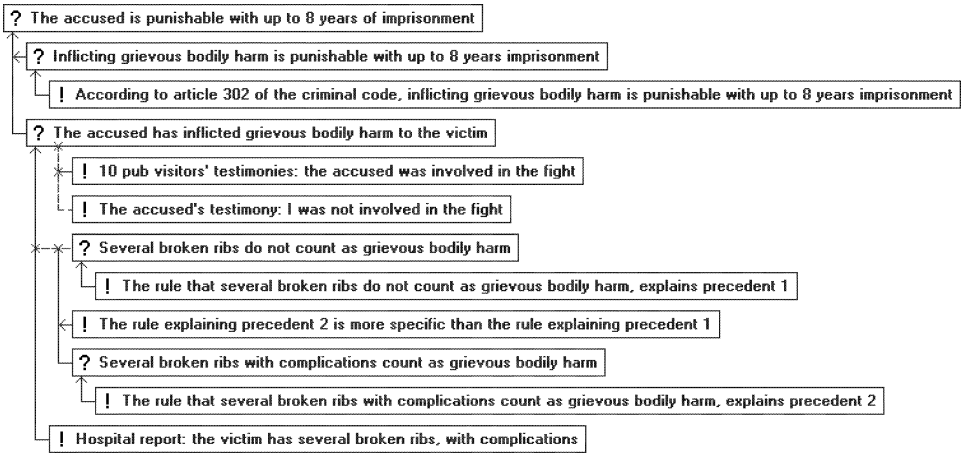

Fig. 20. Attacking that a statement is an undercutter.

no reason justifying the punishability of the accused, but the prima facie reason that the accused has inflicted grievous bodily harm has become unevaluated instead of defeated.

We come to the final piece of information in the case story that could not yet be incorporated in the argumentation: the second precedent that is more on point, and is explained by a more specific rule.[12] The rule explaining precedent 2, viz. that several broken ribs with complications count as grievous bodily harm, has the effect that precedent 1's rule (viz. that several broken ribs do not count as grievous bodily harm) is not defeating. The reason why precedent 2's rule can do this is that it is more specific. The result is shown in Fig. 20. In the end, the conclusion that the accused is punishable with up to 8 years of imprisonment is justified for the reason that he has inflicted grievous bodily harm to the victim.

[12] Precedent-based reasoning in the law has been studied extensively. For instance, Ashley [2] treats the on pointness of cases, and Rissland and Skalak [37]) discuss the use of cases to warrant and to undercut conclusions.

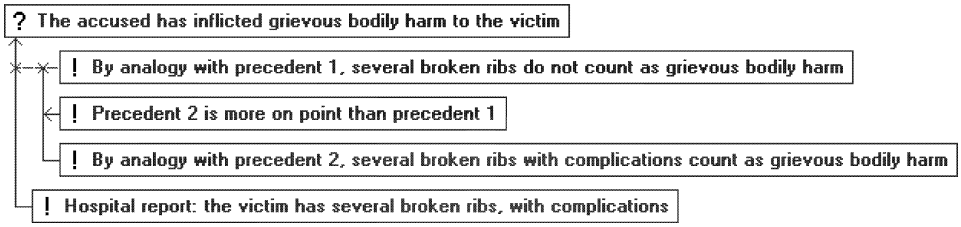

Fig. 21. Attacking that a statement is an undercutter (in terms of on-pointness).

A variant of the precedent-based reasoning is shown in Fig. 21. It makes explicit that precedent 2 is more on point than precedent 1. The argumentation could continue by justifying why this is the case: the reason would be that precedent 2 shares more factors with the current case than precedent 1 since precedent 2 concerns a case of broken ribs with complications.

3.3. Program design

ARGUMED based on DEFLOG uses a 'mouse sensitive' argument screen. Double clicking the screen opens an edit box in which a statement can be typed. Further argumentative data can be added using the context menu that appears after right-clicking the mouse on a statement or an arrow. Recently, a toolbar has been added to ARGUMED based on DEFLOG (Fig. 22). Argument moves can be made by clicking one of the buttons. The toolbar is context-sensitive: only those buttons can be clicked that allow

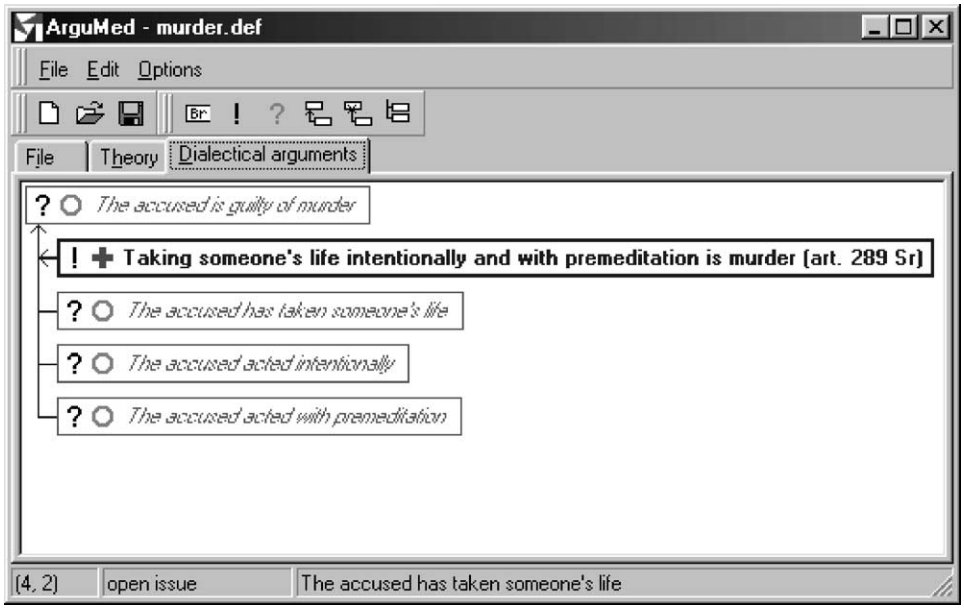

Fig. 22. A conditional statement with a conjunction as antecedent.

moves pertaining to the active statement. For instance, when the active statement is an issue, the 'Set as issue'-button cannot be clicked, while the 'Set as assumption'-button is available. There are buttons for adding an elementary statement, for setting a statement as an assumption or as an issue, to support or attack a statement, and to add a conjunct. Note that use of the buttons in ARGUMED based on DEFLOG and in ARGUE! is different: the latter change the graphical mode (such as the mode of drawing an arrow), while the former correspond to argument moves (such as supporting a statement).

Adding a conjunct to a conditional statement was not yet encountered. Conditionals with conjunctions as antecedents are useful for the representation of rules with composite conditions. For instance, the article on murder in the Dutch criminal code (article 289 Sr) combines three conditions: taking someone's life, intent and premeditation. Fig. 22 shows how this can be represented in ARGUMED based on DEFLOG. The article itself is cited as support for the conditional statement.

In ARGUMED based on DEFLOG, dialectical arguments are computed starting from the conclusion, by recursively adding the reasons for and against the statements in the argument (including the connecting conditionals). When a branch of the argument contains a loop, the recursion stops after the first repeated occurrence of a statement in order to make sure that the resulting graphical structure is finite. The blocking of the recursion is indicated by a series of dots (...). Evaluation occurs automatically in the background. ARGUMED based on DEFLOG computes the dialectical interpretations of the available assumptions, in accordance with the formal definitions of DEFLOG.

When there is more than one dialectical interpretation, each of them can be viewed. Fig. 23 shows two evaluated dialectical arguments corresponding to the two different dialectical interpretations of the same set of assumptions. When there is no dialectical interpretation, all statements are shown as unevaluated (Fig. 24).

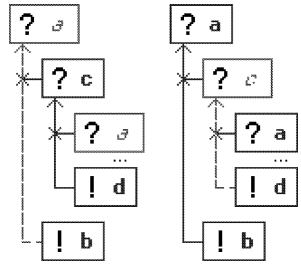

Fig. 23. Two dialectical interpretations.

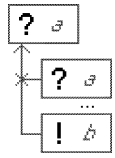

Fig. 24. No dialectical interpretation.

ARGUMED based on DEFLOG has three viewing screens. The first shows the file that contains the argumentative data. It is formatted in an XML-styled format. The second lists the prima facie justified assumptions. The third shows the dialectical arguments as evaluated in accordance with the assumptions' dialectical interpretations. If applicable, the different dialectical interpretations can be viewed by clicking a corresponding dynamically generated button. When there is no dialectical interpretation, this is reported in the status bar, and the dialectical arguments remain unevaluated.

4. Related work

4.1. Argumentation theory

In this subsection, the argumentation theories of ARGUE! and ARGUMED based on DEFLOG, are briefly compared to a selection of theories of defeasible and legal argumentation. ARGUMED 2.0 is the precursor of ARGUMED based on DEFLOG, described by Verheij [47].

As a start, Toulmin's argument scheme [42] is discussed (Fig. 25). Clearly, Toulmin's notions of datum and conclusion have counterparts in all three argumentation theories. Warrant and backing find a natural place in both versions of ARGUMED. Whereas Toulmin only uses warrants for support, ARGUMED 2.0 adds warrants for undercutters, and ARGUMED based on DEFLOG warrants for attack in general. Toulmin's scheme contains rebuttals[13], that just as the defeaters in ARGUE!, the undercutting exceptions in ARGUMED 2.0, and the attacks in ARGUMED based on DEFLOG make argumentation defeasible. The notion of rebuttal is not well elaborated however. A serious omission of Toulmin's work is that he does not discuss argument evaluation. For defeasible arguments, valuation is not a trivial matter, and certainly non-standard. Toulmin's scheme is not put in a procedural context, and does not distinguish between assumptions and issues. The modal qualifier distinguished by Toulmin does not occur in the argumentation theories presented here. Verheij [50] extensively analyzes Toulmin's scheme from the point of view of DEFLOG.

Next, Reiter's default logic [36] deserves discussion, since it can be considered as an early theory of defeasible argumentation. A difference between Reiter's default logic and the present argumentation theories is that the former uses a first-order language with variables and quantifiers, whereas the language of the latter only use elementary sentences (ARGUE!) or sentence connectives (ARGUMED 2.0, ARGUMED based on DEFLOG). The prerequisite α, the justification β and the consequent γ of a default $\alpha : \beta/\gamma$, correspond closely to a reason, the negation of an undercutting exception, and a conclusion, respectively. In DEFLOG, the default would correspond to two sentences, viz. $\alpha \rightsquigarrow \gamma$ and not-$\beta \rightsquigarrow \times(\alpha \rightsquigarrow \gamma)$, where not-$\beta$ is the standard negation of β. Defaults are not conditionals in the logical object language, resulting in the (for long recognized) drawback

[13] Toulmin [42] does not yet make Pollock's distinction between undercutting and rebutting exceptions [28], that is by now standard.

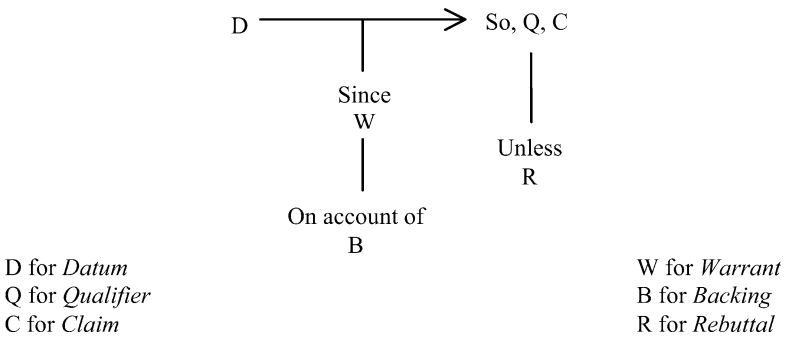

Fig. 25. Toulmin's argument scheme.

that they cannot be derived. This is in contrast with the conditionals of ARGUMED 2.0 and ARGUMED based on DEFLOG. Verheij [48] gives a formal connection between Reiter's default logic and DEFLOG. Reiter's default logic is not put in a procedural context, and does not distinguish between issues and assumptions.

Pollock's theory of defeasible argumentation [28,29] has already been mentioned. Pollock's logical system is richer than the one presented here, for instance, since Pollock adds numerical weights that measure the strengths of reasons. Pollock discusses expressions of the form 'P wouldn't be true unless Q were true', that are closely related to the step warrants of ARGUMED 2.0 and DEFLOG sentences of the form $P \rightsquigarrow Q$. Pollock characterizes undercutters as reasons for the negation of these expressions. This is similar to DEFLOG expressions of the form $U \rightsquigarrow \times(P \rightsquigarrow Q)$, where U is the undercutter. Apparently, there is no discussion of (an analog of) undercutter warrants in Pollock's work. DEFLOG uses a more general kind of attack than Pollock. Both his undercutters and his rebutters can be sensibly expressed in DEFLOG. One way of representing a rebutter R against P as a reason for Q would use sentences of the form $R \rightsquigarrow$ not-Q, $R \rightsquigarrow \times(P \rightsquigarrow Q)$ and $\times(R \rightsquigarrow \text{ not} - Q) \rightsquigarrow \times(R \rightsquigarrow \times(P \rightsquigarrow Q))$. By the first sentence, R is a reason for Q's negation. The second sentence turns R into an undercutter of P as a reason for Q. The third sentence expresses that R is not undercutting when it is not actually a reason for Q's negation (e.g., since it is itself undercut). Pollock's inference graphs (extended with his 'defeat links') are related to the dialectical arguments of the present paper, but are not considered as the analog of classical proofs of a conclusion. Pollock's central use of inference graphs is in the definition of justification. Formal differences are that Pollock's defeat links are a relation on sequents (a supposition-conclusion pair), while ARGUE!'s defeaters work on arguments, ARGUMED's undercutters effect connections between statements and DEFLOG's attacks apply to any statement. Pollock's notion of interests seems to be related to that of issues in the present paper.

In Vreeswijk's abstract argumentation systems [55], the tree-like reason-conclusion structure of arguments (but lacking the coordination of reasons as in CUMULA) is studied in relation to defeat. Vreeswijk uses an (almost) unstructured language with one distinguished sentence that denotes contradiction. He does not include ARGUMED 2.0's

step warrants nor DEFLOG's conditionals expressing support and attack.[14] Vreeswijk considers *inconsistency-triggered defeat* (a term used by Verheij [44]): an argument can only be defeated if there is an undefeated argument with conflicting conclusion. In Vreeswijk's argumentation theory, support and attack are considered separately, viz. in the definition of arguments, and in the definition of the 'in force' arguments, just like in ARGUE!. In ARGUMED 2.0 and DEFLOG, support and attack occur side by side in dialectical arguments. Vreeswijk puts argumentation in a procedural context, but his argumentation sequences have fixed assumptions. Issues are not distinguished. Vreeswijk does not use prima facie justified assumptions as in DEFLOG.

The arguments of Prakken and Sartor's argumentation theory [31] are formed by chaining rules together. Prakken and Sartor's rules are the conditionals of logic programming, and cannot be nested. They are not comparable to ARGUMED 2.0's step warrants since there can be no support for the rules themselves. There are no undercutter warrants. Support and attack are treated separately, and not simultaneously as in the dialectical arguments of the ARGUMED systems. Prakken and Sartor discuss a rebutting and an undercutting type of defeat, where it should be noted that the latter is unrelated to Pollock's standard distinction [28,29]. A naming technique is used for argumentation about priorities. Argumentation is put in a procedural context by the definition of dialogues.

Reason-Based Logic, as initiated by Hage, and further developed in cooperation with Verheij [17,44], can be characterized as a theory of rules and reasons. It does not have a notion of an argument, but focuses on types of sentences related to rules and reasons, and on the states of affairs expressed by sentences of these types. It is of relevance here, since the argumentation theories of the ARGUMED systems have resulted from attempts to bridge the unsatisfactory gap between Reason-Based Logic and CUMULA, as it occurred in my dissertation [44]. Reason-Based Logic's sentences expressing the validity of a rule are comparable to step warrant sentences of ARGUMED 2.0 (or better still to rule sentences supporting them) and similarly to conditional sentences in DEFLOG. Sentences expressing ARGUMED 2.0's undercutter warrants do not occur in Reason-Based Logic, but are related to the validity of a rule with the exclusion of another rule as its conclusion. DEFLOG's attack sentences are related to reasons against a conclusion in Reason-Based Logic. However, in Reason-Based Logic, reasons accrue. The definition of Reiter-style extensions in Reason-Based Logic can be regarded as a definition of the statements justified with respect to a set of assumptions as in ARGUMED 2.0 and DEFLOG. Issues are not distinguished.

Models of precedent-based reasoning like Ashley's HYPO [2] and Aleven's CATO [1] are very different in flavor from the abstract, logic-styled approaches of the present paper (cf. also the logic-styled approaches to precedent-based reasoning by Hage [17], Prakken and Sartor [32] and Bench-Capon and Sartor [6]). However CATO's factor hierarchy is related to the trees of reasons and conclusions as they are used here. Just like the argumentation theories presented here, HYPO and CATO model defeasible reasoning. The focus is on specific kinds of support and attack, related to the role of

[14] In Appendix A, Vreeswijk [55, p. 275ff] briefly discusses Pollock's undercutters by using a richer language which includes defeasible conditionals.

precedents in reasoning. Examples are the analogizing of a precedent, the distinguishing of one precedent from another and the downplaying of distinctions. HYPO represents the dialectical structure of precedent-based reasoning in so-called three-ply arguments.

In Dung's analysis of defeasible argumentation [9], the focus is on an abstract attack relation. DEFLOG is closely related to Dung's argumentation frameworks (see [48,52], where it is shown that essentially Dung uses a less expressive language than DEFLOG). A contrast with all argumentation theories discussed in the present paper is that Dung's analysis does not take the internal structure of arguments into account. Dung does not analyze the connection with support, and considers the attack relation to be fixed: arguing for or against an attack relation is not possible. Dung separates arguments and the attack relation, in contrast with ARGUMED 2.0 and DEFLOG in which support and attack are integrated. Dung's stable extensions correspond to DEFLOG's dialectical interpretations, and Dung's notion of admissibility is related to (but subtly different from) DEFLOG's notion of dialectical justification.

Bondarenko, Dung, Kowalski and Toni [7] have presented an assumption-based framework for default reasoning, related to Dung's work [9]. Assumption-based frameworks are related to DEFLOG (see [48,52]). They are based on a fixed set of rules of inference, in contrast with DEFLOG's use of an object language conditional. Assumption-based frameworks use a contrary mapping that is related to DEFLOG's dialectical negation $\times$ (and *not* to the weak negations that are also used extensively in the assumption-based frameworks). Assumption-based frameworks are mainly presented as a tool to reconstruct nonmonotonic logics (but see [19], where they are applied to legal reasoning).

Summarizing, the present paper's argumentation theories show an innovating development towards the integrated treatment of support and attack in dialectical arguments, the use of object level conditionals expressing support and attack, thereby allowing the expression of warrants and 'anti-warrants' for both supporting and attacking argument steps, and a theory of evaluating dialectical arguments with respect to prima facie justified assumptions. A limitation of the present paper's argumentation theories is their high level of abstraction: specialized kinds of legal reasoning, such as reasoning with precedents, statutes, principles, goals and values have not been analyzed in detail. The argumentation theories, and especially DEFLOG, have been developed as an abstract background against which such dedicated argumentation schemes can be elaborated on (see [51]).

4.2. Argument assistance and mediation

In order to put ARGUE! and ARGUMED based on DEFLOG in context, they are briefly compared to each other and to related systems, viz., Belvedere by Suthers et al. ([41]; for more recent information on Belvedere, see http://lilt.ics.hawaii.edu/lilt/software/belvedere/), Room 5 by Loui et al. [25] and Zeno by Gordon and Karacapilidis [16]. Belvedere is a system to support students engaged in critical discussion of science issues. Room 5 is called a testbed for public interactive semi-formal legal argumentation. Zeno is meant to create advanced support for complex multi-party/multi-goal decision-making. The relation with Verheij's ARGUMED 2.0 [47], the precursor of ARGUMED based on DEFLOG, is also treated. First, the underlying argumentation theories are discussed; second, the user interfaces.

4.2.1. The underlying argumentation theories

In the underlying argumentation theories of all systems argumentation is *dynamic*. Statements can be made, reasons can be adduced, attacks or defeat information can be added. In Room 5 and Zeno, argumentation is *issue-based* as in Rittel's well-known Issue-Based Information System (IBIS) [38]. No new conclusions can be drawn, since these systems focus on the justification of an initial central issue. In Belvedere, ARGUE! and both versions of ARGUMED, argumentation is *free*, in the sense that there is no central issue, and both inference ('forward' argumentation, drawing conclusions from premises) and justification ('backward' argumentation, adducing reasons for issues) are allowed. Also *connecting* previously made arguments (e.g., by turning the conclusion of one argument into a reason for a premise of another argument) is only possible in these systems.

In all systems, reasons can be *chained* (subordination) and can support a conclusion *in parallel* (coordination). In Room 5, Zeno and ARGUMED based on DEFLOG, a distinction is made between *reasons for and against* a conclusion. Belvedere uses typed links, including 'conflicts' and 'explains'. The arguments in ARGUMED 2.0 incorporate counterarguments by means of *undercutting exceptions*. Only ARGUMED 2.0 and ARGUMED based on DEFLOG have notions of the *warrants* underlying argument steps.

All systems model a notion of *defeasible* or *dialectical* argumentation. Belvedere allows the representation of conflicting information. In Zeno, *weighing* the conflicting reasons determines which conclusions are justified. In ARGUE!, support configuration can attack any other support configuration (as long as they graphically fit inside a rectangle). In ARGUMED 2.0, *undercutting exceptions* can block the justification of a conclusion by a reason for it. ARGUMED based on DEFLOG allows the attack of any statement, including the conditionals underlying supporting or attacking argument steps. ARGUE! has *composite-type defeat*, such as defeat by sequential weakening (terms used by Verheij [44]).

In Room 5 and Zeno, argumentation is considered as a *game with participants*. Belvedere is focused on work in small groups. In Room 5 and Zeno, the game character is left *implicit*, but obtained by the distributed access to the systems, on the World-Wide Web. ARGUE! and the two versions of the ARGUMED system, all designed as single-user systems, have no explicit notion of game participants, but can be considered as one-participant games. (The implementation of a multi-participant version of ARGUMED based on DEFLOG is planned.)

Belvedere is only a tool to graphically represent argumentative relations between data. Zeno, ARGUE! and the two versions of the ARGUMED system are *evaluative*: the status of statements and arguments can be determined by the system. In Zeno and the ARGUMED systems, evaluation occurs automatically *in the background*. In ARGUE!, the user *asks* the system to update the evaluation of the statements and arguments. ARGUMED based on DEFLOG uses the logical semantics of DEFLOG for the evaluation of the arguments.

4.2.2. The user interfaces

All systems have a *window-style interface*. Room 5 and Zeno are web applications, ARGUE! and the two versions of ARGUMED are standalone applications that can be run on a PC (downloadable at http://www.rechten.unimaas.nl/metajuridica/verheij/aaa/).

Belvedere and ARGUE! have a *graphical interface*, in the sense that the user draws and organizes the argumentation data on the screen using a pointing device. Room 5, Zeno and ARGUMED 2.0 have a *template-based interface*: users fill in forms to perform an argument move. ARGUMED 2.0 uses different templates for different types of moves. ARGUMED based on DEFLOG uses a mouse-sensitive screen, just like ARGUE!, but the system determines how the argumentative data are organized on the screen.

All systems present arguments in a *visual* manner. Belvedere, Zeno, ARGUE! and the ARGUMED systems use a *tree*-like presentation. Belvedere uses typed links. Room 5 uses a clever system of *boxes-in-boxes* in an attempt to avoid 'pointer-spaghetti'. ARGUMED based on DEFLOG uses trees branching not only at the tree nodes (expressing elementary statements), but also at the tree connections (expressing conditional statements).

In Room 5, Zeno and the ARGUMED systems, counterarguments (based on reasons against conclusions) are *grouped together* in the visual argument structure. Belvedere and ARGUE! leave the organization of the data to the user. In ARGUE!, counterarguments are shown by a *dedicated visual structure*. In the ARGUMED systems, counterarguments are *incorporated* in the arguments themselves, which is possible by the concept of dialectical arguments.

In the ARGUMED systems, the dynamic aspect of argumentation is shown by a *view on the sequence of moves*. In ARGUMED 2.0, it is possible to move back and forth in a line of argumentation. In Belvedere, Room 5, Zeno and ARGUE!, only a *view on the current stage* of the argumentation process is visible. In Room 5 and the ARGUMED systems, it is possible to switch between different views showing different types of information.

5. Conclusion

ARGUE!, the system that was developed first, provides an interesting realization of (and testbed for) a particular theory of defeasible argumentation (a stripped-down version of CUMULA [44]), but that theory is not sufficiently natural to apply to ordinary argumentation. Its user interface, which allows the user to draw and organize argumentative data on screen, is flexible, but also cumbersome due to the complexity of the data structures (especially of the defeaters). As such, it is mainly relevant from a research perspective. For instance, its step-wise evaluation function provides interesting insights into the evaluation of defeasible arguments.

ARGUMED based on DEFLOG is more accessible to ordinary users. ARGUMED based on DEFLOG simplified the warrant model of Verheij's ARGUMED 2.0 [47] by considering the arrows between a reason and its (supported or attacked) conclusion as conditional statements. The result was an expressive and flexible argumentation theory (formalized as the logical system DEFLOG). The evaluation function became logically more satisfactory (with respect to that of ARGUMED 2.0) by its correspondence to DEFLOG. As user interface, a middle way was taken between the too flexible interface of ARGUE! and the too rigid one of ARGUMED 2.0 which used forms. This has been achieved by the use of a mouse-sensitive argument screen in which the argumentative data is organized by the system. Editing the argumentative data occurs directly on the argument screen, instead of in separate templates.

A strong point of the systems is that all allow the evaluation of argumentative data. This is essential for data concerning defeasible arguments and statements. Several other systems stop at the graphical representation of the data (e.g., Belvedere by Suthers et al. [41], and systems based on Toulmin [42], who did not discuss argument evaluation; see [50]). A difficulty is of course that there is no consensus with respect to defeasible argumentation and its (formal) evaluation. DEFLOG has been designed in order to be as transparent as possible with respect to the evaluation of dialectical arguments based on prima facie justified assumptions. Further experience has to be gathered about whether the attempt has succeeded. The test results with ARGUMED are promising in this respect.

Systems using Toulmin's scheme have the advantage that the different slots in the scheme are assigned specific argumentative roles, such as warrant and backing. This can have the effect that a user is forced to better organize his argumentation. A drawback is however that the assigned roles may lead to argumentative rigidity. Further research will have to be done in this direction in order to find out the best approach.

A general question in the design of argument assistants is whether arguments should be graphically represented in the first place. Especially the complexities and subtleties of legal argument may impede such representations, and require natural language representations. (Cf. a recent discussion on the OSSA argumentation theory e-mail list.) A compromise could be the dual representation of arguments, both graphically and in natural language.

Important directions for future research include the integration of domain knowledge and domain-specific argumentation schemes. With respect to the integration of domain knowledge, one can think of diminishing the gap between argument assistants and automated reasoners. Argument assistants have the advantage of being open and flexible, but it is to be expected that the integration of domain knowledge can make the systems more useful in practice. In this way, the advantages of both argument assistants and automated reasoners become available. With respect to the integration of domain-specific argumentation schemes, one can think of typically legal kinds of argumentation, involving precedents, rules, principles, values and goals (of which the understanding has recently increased significantly, mainly by research in artificial intelligence and law). Integrating such schemes opens many difficult questions (such as the way of presenting such involved kinds of arguments), but may increase the usefulness of argument assistants for practical purposes.

With such developments, argument assistants can evolve to valuable knowledge management tools for argument-intensive environments, such as the law.

Acknowledgements

The research reported in this paper was financially supported by the Dutch National Programme Information Technology and Law (ITeR, project number 01437112). The author thanks Jaap Hage, Bram Roth and the editors of this special issue for comments and discussion.

References

[1] V. Aleven, Teaching case-based argumentation through a model and examples, Dissertation, University of Pittsburgh, Pittsburgh, PA, 1997.

[2] K. Ashley, Modeling Legal Argument. Reasoning with Cases and Hypotheticals, MIT Press, Cambridge, MA, 1990.

[3] J. Barwise, J. Etchemendy, Language, Proof, and Logic, Seven Bridges Press, New York, 2000.

[4] T.J.M. Bench-Capon, Argument in artificial intelligence and law, Artif. Intell. Law 5 (1997) 249–261.

[5] T.J.M. Bench-Capon, P.H. Leng, G. Staniford, A computer supported environment for the teaching of legal argument, J. Inform. Law Technol. (JILT) 3 (1998), http://elj.warwick.ac.uk/jilt/98-3/capon.html.

[6] T. Bench-Capon, G. Sartor, Theory based explanation of case law domains, in: Proc. of the 8th International Conference on Artificial Intelligence and Law (ICAIL-2001), St. Louis, MO, ACM, New York, 2001, pp. 12–21.

[7] A. Bondarenko, P.M. Dung, R.A. Kowalski, F. Toni, An abstract, argumentation-theoretic approach to default reasoning, Artificial Intelligence 93 (1997) 63–101.

[8] C.I. Chesñevar, A.G. Maguitman, R.P. Loui, Logical models of argument, ACM Comput. Surv. 32 (4) (2000) 337–383.

[9] P.M. Dung, On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games, Artificial Intelligence 77 (1995) 321–357.

[10] F.H. Eemeren, R. van Grootendorst, T. Kruiger, Argumentatietheorie, Uitgeverij Het Spectrum, Utrecht, 1981.

[11] F.H. Eemeren, R. van Grootendorst, T. Kruiger, Handbook of Argumentation Theory. A Critical Survey of Classical Backgrounds and Modern Studies, Foris Publications, Dordrecht, 1987, Translation of [10].

[12] D.M. Gabbay, C.J. Hogger, J.A. Robinson (Eds.), Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 3, Nonmonotonic Reasoning and Uncertain Reasoning, Clarendon Press, Oxford, 1994.

[13] T.J. van Gelder, The Reason! project, The Skeptic 21 (2) (2001) 9–12.

[14] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: R.A. Kowalski, K.A. Bowen (Eds.), Logic Programming, Proceedings of the Fifth International Conference and Symposium, MIT Press, Cambridge, MA, 1988, pp. 1070–1080.

[15] T.F. Gordon, The pleadings game, in: An Artificial Intelligence Model of Procedural Justice, Kluwer Academic, Dordrecht, 1995.

[16] T.F. Gordon, N. Karacapilidis, The zeno argumentation framework, in: Proc. of the Sixth International Conference on Artificial Intelligence and Law (ICAIL-97), Melbourne, Australia, ACM, New York, 1997, pp. 10–18.

[17] J.C. Hage, Reasoning with Rules. An Essay on Legal Reasoning and its Underlying Logic, Kluwer Academic, Dordrecht, 1997.

[18] J.C. Hage, Dialectical models in artificial intelligence and law, Artif. Intell. Law 8 (2000) 137–172.

[19] R.A. Kowalski, F. Toni, Abstract argumentation, Artif. Intell. Law 4 (1996) 275–296.

[20] R.E. Leenes, Hercules of Karneades. Hard cases in recht en rechtsinformatica, Twente University Press, Enschede, 1998.

[21] A.R. Lodder, DiaLaw—On legal justification and dialog games, Dissertation, Universiteit Maastricht, Maastricht, 1998.

[22] A.R. Lodder, B. Verheij, Opportunities of computer-mediated legal argument in education, in: Proceedings of the BILETA Conference, Dublin, 1998.

[23] A.R. Lodder, A simple model to structure the information of parties in online ADR, in: Proc. of the 8th International Conference on Artificial Intelligence and Law (ICAIL-2001), St. Louis, MO, ACM, New York, 2001, pp. 233–234.

[24] R.P. Loui, Process and policy: Resource-bounded non-demonstrative reasoning, Comput. Intell. 14 (1) (1998) 1–38.

[25] R.P. Loui, J. Norman, J. Altepeter, D. Pinkard, D. Craven, J. Linsday, M. Foltz, Progress on Room 5. A testbed for public interactive semi-formal legal argumentation, in: Proc. of the Sixth International Conference on Artificial Intelligence and Law (ICAIL-97), Melbourne, Australia, ACM, New York, 1997, pp. 207–214.

[26] C.C. Marshall, Representing the structure of a legal argument, in: Proc. of the Second International Conference on Artificial Intelligence and Law (ICAIL-89), Vancouver, BC, ACM, New York, 1989, pp. 121–127.
[27] D. Nute, Defeasible reasoning: A philosophical analysis in Prolog, in: J.H. Fetzer (Ed.), Aspects of Artificial Intelligence, Kluwer Academic, Dordrecht, 1988, pp. 251–288.
[28] J.L. Pollock, Defeasible reasoning, Cognitive Sci. 11 (1987) 481–518.
[29] J.L. Pollock, Cognitive Carpentry: A Blueprint for How to Build a Person, MIT Press, Cambridge, MA, 1995.
[30] H. Prakken, Logical Tools for Modelling Legal Argument. A Study of Defeasible Reasoning in Law, Kluwer Academic, Dordrecht, 1997.
[31] H. Prakken, G. Sartor, A dialectical model of assessing conflicting arguments in legal reasoning, Artif. Intell. Law 4 (1996) 331–368.
[32] H. Prakken, G. Sartor, Modelling reasoning with precedents in a formal dialogue game, Artif. Intell. Law 6 (1998) 231–287.
[33] H. Prakken, G.A.W. Vreeswijk, Logics for defeasible argumentation, in: D.M. Gabbay, F. Guenthner (Eds.), Handbook of Philosophical Logic, Vol. 4, 2nd Edition, Kluwer Academic, Dordrecht, 2002, pp. 218–319.
[34] S. Read, Thinking about Logic. An Introduction to the Philosophy of Logic, Oxford University Press, Oxford, 1995.
[35] C. Reed, D. Walton, Applications of argumentation schemes, in: OSSA 2001: Argumentation and its Applications, The Ontario Society for the Study of Argumentation, 2001.
[36] R. Reiter, A logic for default reasoning, Artificial Intelligence 13 (1980) 81–132.
[37] E.L. Rissland, D.B. Skalak, CABARET: Rule interpretation in a hybrid architecture, Internat. J. Man-Machine Stud. 34 (6) (1991) 839–887.
[38] H.W.J. Rittel, M.M. Webber, Dilemmas in a general theory of planning, Policy Sci. 4 (1973).
[39] G.R. Simari, R.P. Loui, A mathematical treatment of defeasible reasoning and its applications, Artificial Intelligence 53 (1992) 125–157.
[40] A. Stranieri, J. Zeleznikow, Argumentation structures for knowledge management, in: Proceedings of the Third International Conference on the Practical Applications of Knowledge Management (PAKeM-2000), Manchester, The Practical Application Company Ltd, Blackpool, UK, 2000, pp. 51–69.
[41] D. Suthers, A. Weiner, J. Connelly, M. Paolucci, Belvedere: Engaging students in critical discussion of science and public policy issues, in: Proceedings of the 7th World Conference on Artificial Intelligence in Education (AI-Ed 95), Charlottesville, VA, 1995, pp. 266–273.
[42] S.E. Toulmin, The Uses of Argument, Cambridge University Press, Cambridge, 1958.
[43] A. Veerman, Computer-supported collabrative argumentation through argumentation, Dissertation, Universiteit Utrecht, Utrecht, 2000.
[44] B. Verheij, Rules, reasons, arguments. Formal studies of argumentation and defeat, Dissertation, Universiteit Maastricht, 1996.
[45] B. Verheij, ARGUE!—An implemented system for computer-mediated defeasible argumentation, in: H. La Poutré, J. van den Herik (Eds.), NAIC'98. Proceedings of the Tenth Netherlands/Belgium Conference on Artificial Intelligence, CWI, Amsterdam, 1998, pp. 57–66.
[46] B. Verheij, ARGUMED—A template-based argument mediation system for lawyers, in: J.C. Hage, T.J.M. Bench-Capon, A.W. Koers, C.N.J. de Vey Mestdagh, C.A.F.M. Grütters (Eds.), Legal Knowledge Based Systems. JURIX: The Eleventh Conference, Gerard Noodt Instituut, Nijmegen, 1998, pp. 113–130.
[47] B. Verheij, Automated argument assistance for lawyers, in: Proc. of the Seventh International Conference on Artificial Intelligence and Law (ICAIL-99), Oslo, Norway, ACM, New York, 1999, pp. 43–52.
[48] B. Verheij, DEFLOG—A logic of dialectical justification and defeat, Manuscript. See http://www.rechten.unimaas.nl/metajuridica/verheij/publications.htm, 2000.
[49] B. Verheij, Dialectical argumentation as a heuristic for courtroom decision-making, in: P.J. van Koppen, N.H.M. Roos (Eds.), Rationality, Information and Progress in Law and Psychology. Liber Amicorum Hans F. Crombag, Metajuridica Publications, Maastricht, 2000, pp. 203–226.
[50] B. Verheij, Evaluating arguments based on Toulmin's scheme, in: Proc. OSSA 2001: Argumentation and its Applications, The Ontario Society for the Study of Argumentation, 2001.
[51] B. Verheij, Legal decision making as dialectical theory construction with argumentation schemes, in: Proc. of the 8th International Conference on Artificial Intelligence and Law (ICAIL-2001), St. Louis, MO, ACM, New York, 2001, pp. 225–226.

[52] B. Verheij, DEFLOG: On the logical interpretation of prima facie justified assumptions, J. Logic Comput. 13 (3) (2003) 319–346.
[53] B. Verheij, Virtual Arguments. On the Design of Argument Assistants for Lawyers and other Arguers, TMC Asser Press, Hague, submitted for publication.
[54] G. Vreeswijk, IACAS: An implementation of Chisholm's principles of knowledge, in: Dutch/German Workshop on Nonmonotonic Reasoning. Proceedings of the Second Workshop, Universiteit Utrecht, Utrecht, 1995, pp. 225–234.
[55] G. Vreeswijk, Abstract argumentation systems, Artificial Intelligence 90 (1997) 225–279.